Voucher Management System (VMS)

Introduction

Purpose

The functional requirements for the Voucher Management System (VMS) are being discussed with project stakeholders including the Product Development team, IT department, and other relevant personnel involved in its development, implementation, and maintenance.

Scope

The VMS project focuses on managing and redeeming vouchers for FTTH new connections, but does not cover marketing strategies, customer feedback mechanisms, or non-technical aspects of voucher promotions.

Definitions, Acronyms, and Abbreviations

FRD: Functional Requirements Document

FTTH: Fiber-to-the-Home

VMS: Voucher Management System

ERP: Enterprise Resource Planning

SOA: Service-Oriented Architecture

API: Application Programming Interface

SME: Small and Medium Enterprises

References

- IEEE Standard for Software Project Management Plans

- Agile Manifesto

- ISO/IEC 25010:2011 (System and Software Quality Models)

Project Overview

Project Objectives

- Increase promotion and uptake of FTTH services through targeted coupon selling strategies.

- Improve operational efficiency in voucher distribution and redemption processes.

- Enhance customer experience by providing a seamless and user-friendly coupon redemption process.

- Optimize resource utilization by repurposing existing IT solutions for voucher management.

Background and Context

The VMS project aims to improve voucher management and customer satisfaction by leveraging an existing Voucher Management System developed for SME businesses, addressing the current ERP system's lack of flexibility and scalability for effective voucher management.

Stakeholders

Project Sponsor:

Project Manager:

Development Team: Software Engineers, UI/UX Designers

QA Team: Quality Assurance Engineers

End Users: Customer Service Representatives, Sales Team

Compliance Team: Legal and Regulatory Advisors

Project Management

Project Plan

The Voucher Management System (VMS) project will be implemented in four phases for systematic and efficient development and implementation.

Phases

|  |  |
| --- | --- |
| Initiation | The phase focuses on outlining the project scope and objectives to ensure all stakeholders have a clear understanding of the project's objectives. |
| Planning | During this phase, detailed project plans and schedules will be created to outline the project's roadmap and ensure proper resource allocation. |
| Execution | This phase involves executing project tasks, such as developing and testing the VMS. |
| Closure | The final phase focuses on completing the project, delivering the product, and obtaining stakeholder approval. |

Schedule

The project schedule is divided into four phases, each with specific tasks and deadlines, and a detailed Gantt chart will visually represent the timeline and key milestones.

|  |  |
| --- | --- |
| Initiation | Define project scope and objectives.  Identify stakeholders and their roles.  Conduct initial risk assessment.  Prepare project charter. |
| Planning | Develop detailed project plans.  Create project schedule and Gantt chart.  Allocate resources.  Develop risk management plan.  Prepare budget estimates.  Conduct project kickoff meeting. |
| Execution | Develop software according to requirements.  Conduct regular progress reviews.  Perform system integration and testing.  Implement coding standards and source code management.  Ensure data synchronization between ERP and VMS.  Conduct user training and prepare documentation. |
| Closure | Finalize and deliver the VMS product.  Obtain stakeholder approval.  Conduct post-implementation review.  Close project and release resources.  Archive project documentation and lessons learned. |

Monitoring and Control

Weekly status meetings are held to discuss progress, issues, and next steps.

Monthly progress reports provide detailed information on project status, risks, and budget updates.

The Change Management Process is designed to manage any modifications to the project scope or requirements.

Resource Management

|  |  |
| --- | --- |
| Human Resources | Project Manager, Developers, QA Engineers, UI/UX Designers |
| Technical Resources | Development servers, workstations, software licenses |
| Materials | Documentation, training materials |

Risk Management

- \*\*Technical Risks\*\*: Integration issues, performance bottlenecks

- \*\*Management Risks\*\*: Schedule delays, budget overruns

- \*\*Mitigation Strategies\*\*: Regular risk assessments, contingency planning

## 4. Requirements

### Functional Requirements

1. \*\*Voucher Distribution Management\*\*: Manage the sale and distribution of vouchers through the ERP system with necessary configuration adjustments.

2. \*\*Coupon Redemption\*\*: Use the existing VMS to handle coupon redemption for new FTTH connections, ensuring smooth customer onboarding.

3. \*\*VMS Table Parameters\*\*:

- Parent Coupon ID

- Sub Coupon ID

- Coupon Offer Name

- Coupon Activated Date

- Coupon Issued Date

- Coupon Expire Date

- Coupon Face Value

- Sub Coupon Status

4. \*\*Coupon Status Management\*\*: Status transitions and validations:

- Active

- Reserved

- Issued

- Expired

- Cancelled

- Inactive

- Abandoned

5. \*\*Admin Account Capabilities\*\*: Create, modify, and delete records in the VMS table.

6. \*\*Dashboard\*\*: Real-time monitoring and data analysis.

7. \*\*Bulk Data Upload\*\*: Facilitate bulk uploads to the VMS table.

8. \*\*APIs\*\*: Provide APIs for external triggers from SOA, ERP, Cashiering, CRM, etc.

9. \*\*Data Synchronization\*\*: Ensure data synchronization between ERP and VMS.

### Non-functional Requirements

- \*\*Performance\*\*: The system should handle 1000 concurrent users.

- \*\*Security\*\*: Data encryption, user authentication, and authorization.

- \*\*Usability\*\*: Intuitive UI with responsive design.

### Use Cases or User Stories

- \*\*Use Case 1\*\*: As a customer service representative, I want to quickly find and update customer information.

- \*\*Use Case 2\*\*: As a sales manager, I want to generate monthly sales reports to track performance.

## 5. Design

### System Architecture

The system will follow a multi-tier architecture with a presentation layer, business logic layer, and data layer. Components will interact through RESTful APIs.

### Detailed Design

- \*\*Data Structures\*\*: Customer, Order, Product

- \*\*Algorithms\*\*: Data validation, report generation

- \*\*Interfaces\*\*: API endpoints for CRUD operations

### User Interface Design

Mockups and prototypes will include:

- \*\*Login Screen\*\*

- \*\*Dashboard\*\*

- \*\*Customer Profile Management\*\*

## 6. Implementation

### Development Environment

- \*\*Programming Languages\*\*: JavaScript, Python

- \*\*Tools\*\*: Visual Studio Code, Git, Docker

- \*\*Platforms\*\*: AWS for hosting, Jenkins for CI/CD

### Coding Standards

Adopt industry best practices and guidelines, including:

- \*\*Code Readability\*\*: Clear and concise naming conventions

- \*\*Documentation\*\*: Inline comments and API documentation

- \*\*Testing\*\*: Unit tests and code reviews

### Source Code Management

- \*\*Version Control\*\*: Git with GitFlow branching strategy

- \*\*Repository\*\*: Central repository on GitHub

- \*\*Branching\*\*: Feature branches, development branch, main branch

## 7. Testing

### Test Plan

- \*\*Objectives\*\*: Ensure the software meets all requirements and is free of critical defects.

- \*\*Scope\*\*: Functional, non-functional, and regression testing.

### Test Cases

Each test case will include:

- \*\*Input Data\*\*: Specific data sets

- \*\*Expected Results\*\*: Correct outputs

- \*\*Acceptance Criteria\*\*: Conditions for passing the test

### Test Environment

- \*\*Hardware\*\*: Test servers with similar configurations to production

- \*\*Software\*\*: Test versions of all dependencies

- \*\*Network\*\*: Simulated production network conditions

### Defect Management

Defects will be tracked using JIRA:

- \*\*Logging\*\*: Detailed descriptions and steps to reproduce

- \*\*Prioritization\*\*: Based on severity and impact

- \*\*Resolution\*\*: Assignment to developers and follow-up until closure

## 8. Deployment

### Deployment Plan

- \*\*Phases\*\*: Staging, production rollout

- \*\*Strategies\*\*: Blue-green deployment to minimize downtime

### Installation Instructions

- \*\*Pre-requisites\*\*: System requirements and dependencies

- \*\*Steps\*\*: Detailed, step-by-step installation guide

### Rollback Plan

- \*\*Procedures\*\*: Steps to revert to the previous stable version

- \*\*Verification\*\*: Ensuring data integrity post-rollback

## 9. Maintenance and Support

### Maintenance Plan

- \*\*Regular Updates\*\*: Scheduled patches and updates

- \*\*Monitoring\*\*: Continuous system health checks

### Support Plan

- \*\*Support Levels\*\*: Tiered support structure

- \*\*Response Times\*\*: Defined SLAs for different issue severities

- \*\*Contact Information\*\*: Support team contact details

## 10. Appendices

### Glossary

- \*\*API\*\*: Application Programming Interface

- \*\*CI/CD\*\*: Continuous Integration/Continuous Deployment

### Additional Resources

- \*\*Templates\*\*: Project templates and checklists

- \*\*Supplementary Materials\*\*: Additional reading and reference materials